Comparison of Remote Labs in Different Technologies

C. Mergl
National Instruments, Munich, Germany

Abstract— **Recently several possibilities arose to conduct electronic measurement experiments via remote control. Now a comparison of the latest different technologies should bring some answers to interested people, so that they can choose the best technology for them under their criteria. Criteria in this case are, the up-to-dateness of the technology, the development-time, the system-independency of the client in terms of the operating system and internet browser as well as other necessary installations on the client.**

Index Terms— **Remote Control, REL, Virtual Instruments, LabVIEW**

I. INTRODUCTION

Due to the fact that the internet becomes more integrated into our daily lives, several possibilities have arisen to use this cost-effective worldwide standard for distributing data.A lot of different tools are available to publish data from the development environment, like National Instruments LabVIEW, to the Web. Because of the interesting activities in the field of Web-based distance education a lot of remote controlled laboratories are available via the Internet, which are not only unique in their experiments, but also in the tools used to publish them. The purpose of this thesis is to bring some answers to interested people in order to compare the latest technologies which are the LabVIEW built-in Web Server, Nacimientos AppletVIEW, LabVNC and Measurement Studio 8.0.1, which is the only one not supporting LabVIEW as the development environment. These technologies will be compared in terms of the up to-dateness of the technology, the development-time, the system-independency of the client in terms of the operating system and internet browser as well as other necessary installations on the client.

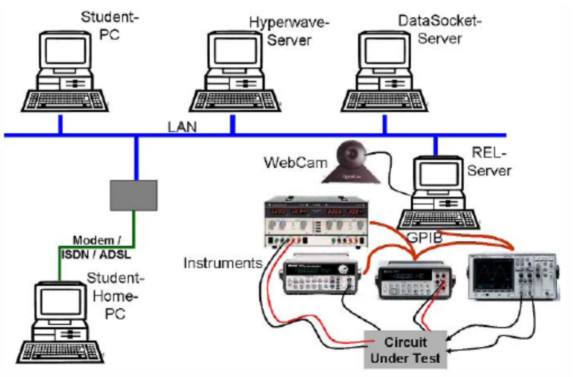


Figure 1 Remote Lab Structure

II. REMOTE CONTROL TECHNOLOGIES

Within this section the state of the art remote control technologies will be presented and shortly explained to get an overview how to work with them. Because LabVIEW 8.20 is used as the development environment for this article, most of these technologies are based on how to pass data between interconnected computers to control LabVIEW VIs remotely via a web browser. These technologies will be the LabVIEW built-in Web Server, a DataSocket Server with ActiveX controls, AppletVIEW which is a commercial product to make remote front panels and LabVNC which is for free. Additionally, as non LabVIEW tool, Measurement Studio 8.0.1, which is a free upgrade to Measurement Studio 8.0.0 will be discussed because of its included suit of Web controls.

A. *LabVIEW Web Server*

With the LabVIEW built-in Web Server it is possible to view and/or control a VI remotely either from LabVIEW or a Web browser. If the VI is controlled by LabVIEW, it must be installed on the client computer. If the VI is controlled via a Web browser only the runtime engine with the same version as within the development computer, so the server, must be installed. This runtime engine consists of a LabVIEW browser plug-in package which will be installed in the mentioned directory of the clients computer. The runtime engine is available for free via the National Instruments Drivers CD or via download from the National Instruments ftp server.
This download is done automatically if a client wants to connect without having the runtime engine already installed. So on the client side, there is the runtime engine and on the server side a HTML file with an <OBJECT> tag, that references to the VI which should become online viewable and controllable[1]. With this tag, the web browser passes the VI to the LabVIEW browser plug-in, because of the information given by this tag, which contains a URL reference to the VI. By typing the Web address of the Web Server into the URL or address field of the clients Web browser, the connection to the Web Server opens. The address contains the computer name or IP address of the server, the port of the web server and the name of the HTML file which references to the VI (e.g. http://nameofserver:80/file.html).

Now the VI will be posted within the client's browser window from the plug-in, which communicates with the Web Server to establish the interaction between client and remote front panel of the VI. Now to control the VI, the client has to request control by selecting "Request Control of the VI", either by right clicking anywhere on the posted front panel where a short cut menu appears, or at the bottom of the Web browser window.

National Instruments recommends using Microsoft Internet Explorer 5.5 with Service Pack 2 or Netscape 4.7 or later on the client side to work with the Web Server. Independent of this the Web Server works with Windows, Linux and Mac operating Systems (there is LabVIEW available for each) and with Internet Explorer, Netscape, Mozilla, Firefox and Opera as Web browser, which are the most used browsers. With other browsers a few changes are necessary to also use them viewing and controlling remote front panels.

The LabVIEW Web Server is available within the LabVIEW Full Development System, the LabVIEW Professional Development System and also as single update. Additionally one can use the LabVIEW Enterprise Connectivity Toolset to add more security features to the Web published VIs.

1) *Web Server Configuration*

To publish VIs on the Web, the Web Server must be enabled, which could be done either with the Web Publishing Tool, or within the "Web Server: Configuration" page which is located in the LabVIEW "Tools" menu at "Options". The Web Publishing Tool, described in the following section, is used to create the HTML files out of the front panel of a VI. Within the "Web Server: Configuration" page you can enable the server, change the root directory, the HTTP port, the timeout and if you want to use a log file or not. For all there are default settings which should work with the most applications, so the default port is 80, the timeout is 60 seconds and the root directory, where the HTML files of the Web Publishing Tool are saved is within the LabVIEW www directory (e.g. C:\Program Files\ National Instruments\LabVIEW 8.2\www).Perhaps in some cases one needs another timeout or has to use another port to communicate, than this is the page where to change.

Under the "Web Server: Visible VIs" page of the Options dialog box you can decide whether a VI is visible or invisible on the Web. Therefore a list of VIs can be created where it is possible to allow or deny access to this VIs or even groups of VIs or directories. By default all front panels are visible, so for each the access is allowed. If a client wants to connect to a published VI the Web Server checks the list of the VIs within the "Web Server: Visible VIs" page, compares it with the client request and if the VI is on the list, allows or denies access depending on the respective setting.

Not only the visibility of the VIs could be set, furthermore it is also possible to configure which browser address (client address) can view and or control the published VIs. This is done by creating a Browser Access List within the "Web Server: Browser Access" page, also within the Options dialog box. After a client attempts to connect, the Web Server compares its address with the Browser Access List and determines if access is permit or denied. It is possible to set for each "Allow Viewing and Controlling", to only "Allow Viewing" and to "Deny Access", at which by default everyone has access and is allowed to view and control the published VIs. All other Configurations of the Options dialog box, selected within the Tools menu, are presented quite similarly and can be accessed by selecting them out of the Category list on the left side of the dialog box. A few of these settings can also be changed within the Web Publishing Tool, which will be discussed within the next section.

2) *Web Publishing Tool*

Until yet the Web Server has only been discussed in general terms like how it works, which configuration must or could be done and so on. Now we are talking about how to make a VI remotely controllable.

The Web Publishing Tool is a LabVIEW built-in tool to publish the front panel of a VI as a HTML document to the web. Accessible also within the Tools menu of LabVIEW, there are three steps from choosing a VI till saving the HTML file to disk. In the first step "Select VI and Viewing Options" the VI to publish, which must be in memory, has to be selected. Also the "Viewing Mode" can be changed between "Embedded", "Snapshot" and "Monitor", where Embedded allows clients to view and control the front panel, Snapshot to only display a static image of the front panel and Monitor to display a snapshot with a configurable updating interval.

For the next step "Select HTML Output" you can type in a title (document title), a text before (header) the front panel and a text after (footer) the front panel which is going to be displayed on the respective place.

The next, and also last step of the procedure is the "Save the New Web Page" step where the created HTML file of the VI is going to be saved to a directory (by default this will be …\LabVIEW8.2\www) with the selected filename (e.g. PublishedVI.html) and a URL will be created (e.g. http://computername:80/PublishedVI.html). After saving it, the VI now is ready to be remote controlled from a client by typing this URL into the address or URL field of his Web browser window.

3) *Web Server Conclusion*

So with the LabVIEW built-in Web Server it is possible to remote control VIs from clients who have at least the runtime engine installed on their systems. Also different settings according the configuration of the Web Server within the Options dialog box can be done. The creation of a Web site with remote front panel could easily be

done with the Web Publishing Tool, so a HTML file is accessible for a client, created out of a VI, at which no modification of the code is necessary.
National Instruments recommends publishing data-intensive VIs to be remote controlled, which means for example VIs with several charts or while loops running without a wait function to give other tasks time to perform [2]. This recommendation is not always possible to follow by developing applications, what should be kept in mind by programming VIs to publish them on the Web.

B. *DataSocket Server*

DataSocket is based on TCP/IP and enables live data exchange between different applications, running on the same computer or also on different computers of a network [3]. So consisting of two components, the DataSocket API and the DataSocket Server, DataSocket is a high-performance programming interface for sharing and publishing data in different applications, on different systems, connecting different I/O technologies. The DataSocket API is the interface for communicating with multiple data types and languages; the DataSocket Server does the Internet communication in form of TCP/IP programming.
Instead of using TCP/IP where one has to write code to convert the data to a stream of bytes and also to pars it back into the original form in the subscribing application, DataSocket transfers the data in an unlimited number of formats, including strings, Booleans and also waveforms. So in the same way a Web browser pulls together different Internet technologies like HTTP, HTTPS, FTP and FILE, DataSocket pulls together different communication protocols.

1) *DataSocket API*

The DataSocket API, based on URLs for connecting to the data, is protocol-independent, language-independent and OS-independent, and publishes binary data. This API is implemented in LabVIEW as ActiveX Control, as LabWindows/CVI C library and as a couple of LabVIEW VIs, to use it in any development environment. So the API converts the data into a stream of bytes to send it across the network, where on the other end the subscribing DataSocket application converts this stream back to its original form, which eliminates the complexity of the normally written code by using TCP/IP libraries instead.
Consisting of four basic actions (open, read, write, close), the DataSocket API can be used for data items on HTTP, FTP and DSTP Servers, and also for local files and OPC Servers. 1.

2) *DataSocket Server*

The DataSocket Server is a stand-alone component, with which data can be broadcasted across the Internet to several clients concurrently, using the DataSocket API. The connections to the clients are managed automatically which simplifies network TCP programming [4].Broadcasting data, using the DataSocket Server requires a publisher, the DataSocket Server itself and a subscriber. The date will be written, using the DataSocket API, to the server from the publisher. Also the subscribing application uses the DataSocket API to read the data from the server. So both, the publisher and the subscriber, are clients of the DataSocket Server and also can reside on the same machine, but normally are running on different machines. The following figure 2 represents how two clients, where both are publisher and subscriber, works for controlling a VI remotely with LabVIEW and the Internet Explorer as clients.

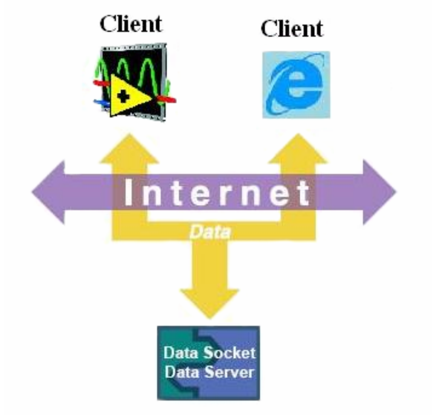


Figure 2 DataSocket Server with LabVIEW and Internet Explorer as Clients

3) *Using DataSocket for remote controlling VIs*

As discussed before, and also illustrated in figure 2, there are two clients, where both are publishers and subscribers using the DataSocket Server for sharing data via the Internet. So on one side there is LabVIEW which is able to communicate with a DataSocket Server by using the DataSocket API VIs and on the other side there is the Internet Explorer. Consider the LabVIEW Web Server where the front panel of the VI is published within the Web browser window, it is contrary using DataSocket Server, because the LabVIEW application don’t needs a front panel in form of a user interface. The published user interface for controlling the VI via DataSocket could be construct e.g. with Visual Basic. Therefore in Visual Basic ActiveX Controls has to be selected from the respective DataSocket, which then can be used to build the user interface. To get a user interface out of the ActiveX controls, it is necessary to create an Internet distribution package.

4) DataSocket Conclusion

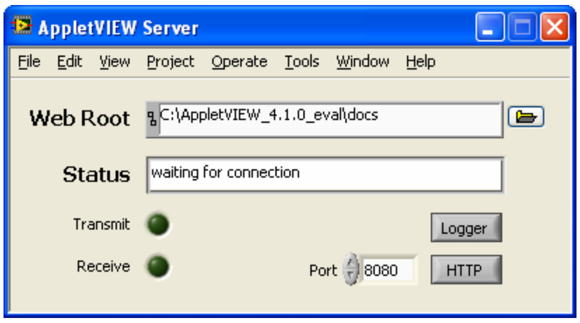
So as one client there is LabVIEW publishing the measured data and acting in order to the ActiveX controls of the Web browser, how subscribes the data from LabVIEW and shows it within the Web Browser window. Instead of the runtime engine, which has to be installed on the client controlling the VI via a Internet browser, here a package of cabinet files, which are the user interface within the browser and the DataSocket server must be installed on this client. As discussed before, DataSocket is quite independent, but because of using the ActiveX controls this solution is bound to use Windows operating systems on the client side.
The LabVIEW VIs must be build using the DataSocket API functions to communicate with the DataSocket Server, which could be seen as a modification to the normal code for controlling instruments and taking measurements.

C. AppletVIEW

AppletVIEW is a toolkit for LabVIEW from Nacimiento Software Corporation which works in terms of development of a remote controlled VI quite similar to the LabVIEW Web Server.
The AppletVIEW toolkit uses the Java and TCP/IP to remote control a VI from any standard Web browser [5]. Because using Java and TCP/IP it is also quite independent from Web browsers or operating systems, used at the client side.
The latest Version, AppletVIEW 4.0 is compatible with LabVIEW 6.x and 7.x, and Java 1.1 through 1.5, it is using standard HTTP and is also compatible with all firewalls and proxy servers. So AppletVIEW enables the control of LabVIEW VIs to be controlled remotely using Java applets in standard Web browsers, where no plug-ins, modifications of the LabVIEW code or the LabVIEW runtime engine, on the client system must be installed.
The Java applet used by AppletVIEW is compliant with Java 1.1 and can run in any Web browser, supporting Java 1.1 [6]. As examples for the minimum versions of Web browser to use on the client side, Nacimiento recommends to use he Netscape Communicator 4.05 or higher and the Microsoft Internet Explorer 4.01 or higher to use it with Windows 9x, Windows NT 4.0, Windows 2000 and also Unix operating systems. By using Mac operating systems, Netscape Communicator 4.76 and the Microsoft Internet Explorer 4.0 or higher is recommended. Generally, every browser able to use Java 1.1 or higher, is able to remotely control VIs published using AppletVIEW. Currently the latest Version of AppletVIEW (4.0) is only supporting LabVIEW 7.x as latest version of LabVIEW and it is unknown when a new version, supporting LabVIEW 8.x will be available.

1) Process of publishing VIs with AppletVIEW

As the first step the AppletVIEW Server has to be opened, which is used to publish a LabVIEW VI to the Internet. The following figure 3 represents the AppletVIEW Server.



Web Root C:\AppletVIEW_4.1.0_eval\docs
Status waiting for connection
Port 8080

Figure 3 AppletVIEW Server

The AppletVIEW Server can either run as a stand-alone application, as a top level window, or can be placed within the applications diagram [7]. The next step is to open the AppletBuilder, which is written in Java, and is necessary to create a Java applet from the LabVIEW application.
The AppletBuilder is for reading the front panel of the VI to publish and recreate it in Java. This is done by selecting the system where the AppletVIEW Server is running, which is normally the localhost, to show a list of the VIs in memory where now the VI to publish must be selected. AppletBuilder is able to read most of the information about the front panel of a LabVIEW VI, but usually there are some modifications necessary to let it appear in the same form it is represented within LabVIEW. It is possible to resize, move and change properties of all components within the AppletBuilder per clicking and dragging, but not all the controls are available in Java as they are appearing in LabVIEW. So instead of a stop button in this applet a kind of switch is represented, as well as instead of a rectangular LED a round one appears. Now this Java front panel must be saved as a .vmil file. Within the AppletBuilder it is also possible to change between edit mode and run mode, to also be able to test the published VI while creating the applet. Now to create the HTML file for the applet the code is generated in the AppletBuilder by choosing File>HTML. This code must be copied into a text editor window and saved as .html file. This file now could be opened via the internet to remote control the published VI via an ordinary Internet browser meeting the requirements in form of Java enable.

2) *AppletVIEW Conclusion*

So AppletVIEW is the first solution, publishing VIs, where now download for the client is necessary to remote control it via the Internet, because nowadays every Internet browser is able to use Java applets.
The only constraint is that there is no current version available supporting LabVIEW 8. So one, who is developing with LabVIEW 8 has to save all his files for the current version 7.1 and than form 7.1 to 7.0 to be able to use AppletVIEW to publish his VIs.
Also the applet is not identical with the front panel of the VI, so that there are always little changes necessary, but on the code itself are no further modifications necessary.
Per default AppletVIEW is using port 8080 for HTTP and VITP runs on port 4749, but both could be changed by changing the server configuration and the applet tags [8].

D. *LabVNC*

LabVNC is also quite similar in using it to publish VIs to the LabVIEW Web Server and Nacimientos AppletVIEW. It is a open source utility that uses the open-source VNC protocol to turn any VI into a Java applet to remote control it via the net [9].Using LabVNC to publish VIs also needs no further modification of the published VIs. The current version is LabVNC BETA 0.4 which needs the following system requirements:

Server: LabVIEW 6.0 or higher, Windows 95/98/NT/2000
Client: Any Java-enabled Web browser on any operating system

LabVNC consists of a VI control panel, some DLLs, and a binary executable that is the LabVNC server. With the actual version the server only runs on 32-bit Windows. LabVIEW 6.0 or higher must be installed on the server machine. The remote clients can be on any platform (Windows, Linux, MacOS, etc.) for which there is a Java-enabled browser.

1) *Publishing VIs using LabVNC*

For publishing VIs using LabVNC the LabVNC.vi has to be opened, which is illustrated in the following figure 5.

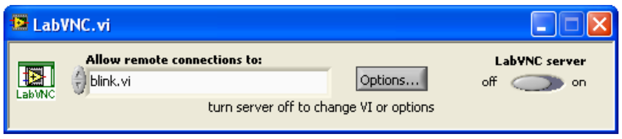

LabVNC.vi
Allow remote connections to:
blink.vi
Options...
LabVNC server
off
on
turn server off to change VI or options
LabVNC

Figure 5 LabVNC.vi

Under “Allow remote connections to:” the VI to publish, which must be in memory, must be choosen. After turning the “LabVNC server” switch to on, after a few seconds a tray icon in the windows tray shows that the server is running. Now the server is ready to accept remote connections. The first time the server runs, a password for the clients need to be set. The client needs to open a Internet browser and point the URL to: http://<machine name or IP address of server>:5800/

2) *LabVNC Conclusion*

LabVNC is a quite old open source utility, based on Windows 9.x, Windows NT and Windows 2000 operating systems, using LabVIEW 6.0 or 7.0 as development environment.
There is no new version available which might support LabVIEW 8.x on a windows XP platform, which does not work with the actual version of LabVNC. So it is nice to have a freeware tool for publishing VIs, which needs no download to use it on the client side, but on the server side older development environments and also operating systems must be used to publish VIs using LabVNC.

E. *Measurement Studio 8.0.1*

National Instruments Measurement Studio uses the concept of virtual instrumentation to deliver measurement and automation programming tools to a Visual Studio application development productivity by integrating a suite of tools and class libraries for Visual Studio [10]. Measurement Studio 8.0.1 is a free upgrade to Measurement Studio 8.0, which fully supports Visual Studio. This free upgrade supports version 2.0 of the .NET Framework and MFC 8.0 in Visual Studio 2005 applications. Measurement Studio 8.0.1 includes a complete suite of ASP.NET Web controls to build remote monitoring and control applications. The so created Web pages do not require any runtime engine on the client, thus they can be displayed on any operating system, any browser and any client.
This software has nothing to do with LabVIEW as development environment, but because of the actuality and the possibility to be totally independent from any downloads, operating systems or browser software of the client it is presented also within this article.

III. COMPARISON OF THE TECHNOLOGIES

Within this section, the previously discussed technologies to remotely control an electronic laboratory developed with LabVIEW will be discussed in different terms like development time and user handling. The last presented technology, which is Measurement Studio 8.0.1 will not be compared with the others because of Visual Studio as development environment in this case.

A. *Development*

Now all four discussed technologies, which are the LabVIEW Web Server, DataSocket Server, AppletVIEW and LabVNC, to publish LabVIEW front panels to the Web, are using LabVIEW as development environment. All these technologies, excepting the DataSocket Server need no modification of the LabVIEW code, besides using AppletVIEW a few changes in developing the published applet of the front panel has to be done.
So using LabVIEW Web Server and LabVNC, the published front panel within the clients Internet browser window looks the same as it does in LabVIEW itself. Using AppletVIEW a few changes with the AppletBuilder must be done. Using the DataSocket Server, the LabVIEW front panel is completely independent from the one presented in the clients browser, because this front panel must be developed using e.g ActiveX controls within Visual Basic. So using the DataSocket Server, you have to construct all the VIs for controlling a remote laboratory always with having in mind that the necessary data must be sent to the DataSocket Server via the DataSocket API VIs and also received in the same way. This means a lot of additional code and also another general construct of the code. While using e.g. the LabVIEW Web Server, the VI could be constructed as it works on the development PC, using DataSocket Server the code becomes much more complex.
Furthermore AppletVIEW and LabVNC does not work with LabVIEW 8.2 as development environment, which causes to use an older version like LabVIEW 7 to construct the VIs. Also there will be no new version of LabVNC and it is unknown if there will be a new one of AppletVIEW, to use them with the latest version of LabVIEW.
The LabVIEW Web Server and LabVNC needs the smallest development time because there are no additionally work to do using them. Using AppletVIEW the development time increases a little bit because of the changes which had to be done within the AppletBuilder and the construction of the HTML page, which is done automatically using the Web Server or LabVNC. With DataSocket Server used to publish VIs, even the construction of the LabVIEW code needs more time and additionally the front panel has to be created using another software, like Visual Basic, which overall needs much more development time.

B. *Client Requirements*

The needs of the client for working with the published VIs are also a necessary aspect that must be discussed independent of the development or the server itself. So within the following subchapters, the different technologies will be discussed concerning the client requirements and they will be compared with each other.

1) *LabVIEW Web Server*

Using the LabVIEW Web Server to publish VIs to be remotely controlled via the Internet, the client needs to install the LabVIEW runtime engine which is about 23MB. The WebServer is quite independent form the clients operating system or Web browser. Every operating system, for which a runtime engine, so LabVIEW, is available, could be used with every standard Web browser.
The Web Server works with Windows, Linux and Mac operating Systems (there is LabVIEW available for each) and with Internet Explorer, Netscape, Mozilla, Firefox and Opera as Web browser, which are the most used browsers

2) *DataSocket Server / ActiveX Controls*

Because of using ActiveX controls on the front panel within the clients browser window to communicate with the DataSocket Server, it is necessary to have a Windows operating system. ActiveX is a Microsoft technology for data exchange between different applications (e.g. Excel table in Word), which is not supported from other operating systems like Linux or Mac.
Also the client needs to install a few .dlls and .cab files, which could be seen as a plug-in for the Web browser.

3) *AppletVIEW*

Because AppletVIEW is using Java 1.1 the published front panel can run in any Web browser supporting Java 1.1. As examples for the minimum versions of Web browser to use on the client side, Nacimiento recommends to use he Netscape Communicator 4.05 or higher and the Microsoft Internet Explorer 4.01 or higher to use it with Windows 9x, Windows NT 4.0, Windows 2000 and also Unix operating systems. If using Mac operating systems, Netscape Communicator 4.76 and the Microsoft Internet Explorer 4.0 or higher are recommended. Generally, every browser able to use Java 1.1 or higher, is able to remotely control VIs published using AppletVIEW. So using AppletVIEW, normally no further installations must be done on the Client side, because nearly everybody has Java already installed because of surfing on websites using Java applets.

4) *LabVNC*

On the Client side LabVNC is really independent from any operating system or Web browser.
Only a Java-enabled Web browser is necessary, which is almost in included with every OS.

AppletVIEW and LabVNC crystallise as the best technologies to use, to become completely independent as client. But remember, they are also the ones which are not able to use LabVIEW 8.2 as development system.

C. *User Handling*

By developing a remote laboratory, where instruments are controlled, different measurements and experiments could be done a user handling is necessary, in form of a log in and a timing to give every user a defined time to work with it.
Using the LabVIEW Web Server it is very easy to implement, because the timing could be constructed as the VI runs only on the development system. The connection to the clients could be closed programmatically using properties of the Web Server. If a client wants to work with a Web Server build, published VI he has to request control by clicking the right mouse button within the front panel and selecting it from the appearing menu. After finishing the control is getting released because of the programmatically closing of the connection to the client. The timing settings are also possible to set via the Web Server configuration where a defined value is given to release the control of the currently controlling client if another client connects, where he is automatically in viewing mode. After this time is over, the viewing client gets the control.
Using the DataSocket Server this all has to be done programmatically in LabVIEW which is a little bit more complex, because of not being able to use the Web Servers properties.
With AppletVIEW it is quite similar to the LabVIEW Web Server, but it has to be constructed with the AppletVIEW VIs instead of property nodes. There are several VIs to check connections to clients, so VIs which handles each HTTP request, the interface between a remote Web client and an open VI and also one VI to close the connection to the client. This VI is the Close Applet Connection.vi which closes the connection to the client when called. But there is now possibility to automatically give the control to next viewing client, as it could be done with LabVIEW Web Server.
With LabVNC there is no user handling in terms of request/release control of the VI, in terms of closing connections to the clients available. So in terms of the necessary user handling, LabVNC is inappropriate to develop a remote control of a laboratory.

D. *Conclusion*

Now after comparing the different technologies in terms of development time, client requirements and user handling many advantages and disadvantages arose, which are represented within the following.

a) *LabVIEW Web Server*
    + shortest development time
    + best user handling of all
    + client runs on Windows, Linux, Mac and with nearly every browser
    - runtime engine needed to install on client

b) *AppletVIEW*
    + short development time
    + user handling possible
    + no client requirements
    + operating system and browser independent
    - only available for LabVIEW 6.0 and 7.0 for development

c) *LabVNC*
    + short development time
    + no client requirements
    + operating system and browser independent
    - no user handling
    - only available for LabVIEW 6.0 and 7.0 on Windows 9.x, NT and 2000 OS

d) *DataSocket Server*
    + available for LabVIEW 6.0 to latest
    + user handling possible
    - high development time
    - only Windows clients can use
    - installation of files needed on client

Because of using LabVIEW 8.20 as the development environment the best solution to use is the built in LabVIEW Web Server for publishing VIs. The construction of the code can be done as the VI would run locally, no additional things are necessary. With the Web publishing tool it is very easy and fast to build a HTML page of the VI to remote control. Also a lot of additional possibilities like the user handling, timeouts and so on are implemented, which could be set within the configuration or programmatically in LabVIEW. LabVNC and AppletVIEW are not really up to date because of the development four to five years ago. The DataSocket technology is quite useful, but not to construct a remote laboratory, alone because of the constraint to use a Windows operating system on the client side.

IV. CONCLUSIONS

The current technologies to publish VIs on the Web, which are the LabVIEW built-in Web Server, Nacimientos AppletVIEW, LabVNC and Measurement Studio 8.0.1, have been presented and discussed in terms of their features and have been compared according to different criteria. These criteria are the up to-dateness of the technology, the development-time, the system-independency of the client in terms of the operating system and internet browser as well as other necessary installations on the client.
The comparison of the different technologies reflected that the LabVIEW built-in Web Server is the current best technology to use. The construction of the code can be done as if the VI was running locally, no additional

modifications are necessary. With the Web publishing tool it is very easy and fast to build a HTML page of the VI to remote control. Also a lot of additional possibilities like the user handling, timeouts and so on are implemented, which could be set within the configuration or programmatically in LabVIEW.

REFERENCES

[1] LabVIEW Basics II: Development – Course Manual National Instruments, March 2004 Edition, Part Number 320629M-01, pp.7-21

[2] LabVIEW Basics II: Development – Course Manual National Instruments, March 2004 Edition, Part Number 320629M-01, pp.7-21

[3][4] Integrating the Internet into your Measurement System National Instruments, March 1999 Edition, pp.4

[5] www.appletview.com/index.html, Web site of AppletVIEW Toolkit Nacimiento Software Corporation 2004

[6] www.appletview.com/browser.html, Web site of AppletVIEW Toolkit Nacimiento Software Corporation 2004

[7] www.appletview.com/tour.html, Web site of AppletVIEW Toolkit Nacimiento Software Corporation 2004

[8] AppletVIEW User Manual, Nacimiento Software Corporation, pp.12-1

[9] http://www.jeffreytravis.com/lost/labvnc.html, Web site of LabVNC Jeffrey Travis Studios LLC 2004

[10] http://sine.ni.com/nips/cds/view/p/lang/en/nid/3769, Web site of NI Measurement Studio, National Instruments 2006

AUTHOR

C. Mergl is Applications Engineer at National Instruments, Munich (e-mail: Christian.Mergl@ni.com).